研究与开发

基于拟态防御架构的服务功能链执行体动态调度方法

李传煌，唐晶晶，陈泱婷，雷睿，陈超，王伟明
（浙江工商大学信息与电子工程学院（萨塞克斯人工智能学院），浙江 杭州 310018）

摘　要：面对静态、滞后的传统防御技术无法有效应对新型网络攻击的问题，根据拟态安全防御理论，提出了一种建立在数据转发层面的拟态服务功能链（mimic service function chain，MSFC）防御架构，基于该架构进一步提出了一种基于判决反馈的执行体动态调度方法。该方法以判决器反馈的异常执行体信息、执行体的异构度以及系统的实际负载量作为调度影响因素，使调度方法可以根据网络实际变化进行自适应调整。此外，该调度方法利用判决反馈对调度时间进行调整，以达到系统花费与安全性的最佳平衡，降低了系统的资源开销。仿真结果表明，该调度方法可以在平衡系统花费与安全性的基础上，选出更符合当前网络需求的高异构度执行体集合，从而提升系统的安全性及可靠性。

关键词：服务功能链；拟态防御；动态调度；执行体；异构度
中图分类号：TP393
文献标志码：A
doi: 10.11959/j.issn.1000−0801.2022070

Dynamic scheduling method of service function chain executors based on the mimic defense architecture

LI Chuanhuang,TANG Jingjing, CHEN Yangting, LEI Rui, CHEN Chao, WANG Weiming
School of Information and Electronic Engineering (Sussex Artificial Intelligence Institute),
Zhejiang Gongshang University, Hangzhou 310018, China

Abstract: Faced with the problem that static and lagging traditional defense technologies cannot effectively deal with new network attacks, according to the theory of mimetic security defense, a defense architecture of mimic service function chain (MSFC) based on the data forwarding level was proposed, and an execution dynamic scheduling method based on the decision feedback was further proposed. The method took the abnormal executor information fed back by the decision maker, the heterogeneity of executors and the actual load of the system as the scheduling influencing factors, so that the scheduling method can be adjusted adaptively according to the actual changes of the network. In addition, the scheduling method used decision feedback to adjust the scheduling time, so as to achieve the best balance between system cost and security, and reduce the resource overhead of the system.

收稿日期：2022−01−27；**修回日期**：2022−04−06
通信作者：李传煌，chuanhuang_li@zjgsu.edu.cn
基金项目：国家自然科学基金资助项目（No.61871468，No.62111540270）；浙江省新型网络标准与应用技术重点实验室资助项目（No.2013E10012）
Foundation Items: The National Natural Science Foundation of China (No.61871468, No.62111540270), Zhejiang Key Laboratory of Network Standards and Applied Technology (No.2013E10012)

Simulation results showed that the scheduling method can select a set of highly heterogeneous actuators that better meet the current network requirements on the basis of balancing the system cost and security, so as to improve the security and reliability of the system.

Key words: service function chain, mimic defense, dynamic scheduling, executor, heterogeneous degree

0 引言

近年来，互联网行业高速发展，网络技术被应用在金融、政务、交通等诸多领域。但互联网给生产生活带来极大便利的同时也带来了许多安全问题，形式多样的网络攻击层出不穷，分布式拒绝服务（distributed denial of service，DDoS）攻击、数据泄露以及渗透攻击等新型的网络攻击对提供计算、存储、网络服务的云计算等虚拟化网络环境构成了巨大的威胁。因此，如何依据目前的网络威胁有针对性地构建新型安全防御方案成为了亟须解决的问题。

端到端服务的交付通常需要多种服务功能（service function，SF），包括传统的网络服务功能（如防火墙、IP 网络地址转换器等）及特定于应用程序的服务功能。服务功能链（service function chain，SFC）指 SF 有序排列的集合，它能够处理分类后的数据包、数据帧或数据流[1]。传统网络中的 SFC 与物理拓扑的紧密连接造成 SFC 变更时将产生较高的网络开销与潜在的利润损失[2]，而软件定义网络（software defined network，SDN）[3-5]和网络功能虚拟化（network functions virtualization，NFV）[6-8]技术的运用使上述问题得以方便解决，SFC 再次成为当前研究和应用的热点[9]。NFV 技术构建各种虚拟 SF 资源池，SDN 技术用控制转发分离的方式动态集中调度流量，进而实现定制化和灵活化的对接。

随着基于 SDN 和 NFV 技术的 SFC 使用量增涨，其安全性的需求日益增大。SF 是 SFC 的关键组成部分，SFC 域中的所有数据流必须经过特定的 SF。如果 SF 被控制，攻击者可以通过被挟持的 SF 随意篡改、丢弃数据，甚至通过拒绝服务（denial of service，DoS）攻击[10]使服务功能转发器（service function forwarder，SFF）瘫痪[11]，不但 SF 不能完成指定功能，且整个 SFC 域可能面临崩溃。本文针对 SFC 中的 SF 安全问题，根据拟态安全防御理论中的动态异构冗余（dynamic heterogeneous redundancy, DHR）模型[12]，提出了一种建立在数据转发层面的 MSFC 防御架构。

拟态安全防御类似生物的拟态伪装，防御者通过控制网络、运行环境、软/硬件组成等元素的动态变化以达到自身不易被成功攻击的目的。拟态安全防御[13]（mimic security defense, MSD）是在主动或被动的触发条件下，动态随机地选择多个执行对应功能的硬件变体和软件变体。因此，系统具备动态性、随机性、冗余性、异构性以及非持续性等特点。系统的上述特点使内部攻击者和外部攻击者都无法准确获取系统内部元件的运行环境以及状态信息，因此无法根据后门和漏洞建立稳定的攻击链，从而达到提升系统安全性的目的。所以，拟态安全防御本质是一种结合被动防御与主动防御的新型防御体系。目前的拟态研究，包括拟态控制器[14-15]、拟态 Web 服务器等在控制层面建立拟态构架，而在数据转发层面的研究较少。

本文在提出的 MSFC 防御架构的基础上，进一步提出一种基于判决反馈的动态调度方法，以判决器反馈的异常执行体信息、执行体的异构度以及系统的实际负载量作为调度影响因素，使调度方法可以根据网络变化进行自适应调整，从而提升系统的安全性。此外，该调度方法基于判决反馈对调度时间进行调整，以达到系统花费与安全性的最佳平衡，降低系统的资源开销。

1　相关研究现状

在 SFC 安全研究方面，现有的防御方案总结见表 1。

表 1　防御方案总结

防御机制	防御方案	特点
被动防御	SF 安全加固	需增设安全模块，造成额外开销
	SFC 安全编排与部署	未知漏洞易被攻破
主动防御	移动目标防御	计算难度较大
	拟态安全防御	对调度算法质量要求较高

被动防御主要有 SF 安全加固和 SFC 安全编排与部署两类方案。SF 安全加固方法通过增设安全模块降低风险。文献[16]提出了一种验证 SFC 中策略是否正确执行的安全策略分析模型，并基于该安全模型设计了一种可以应对同一 SFC 中 SF 的错误交互导致的安全漏洞等网络潜在威胁的软件工具，但该方案同时将额外造成大量的时间与资源开销。SFC 安全编排与部署方法通过对 SF 的逻辑结构、部署方式等进行调整提升安全性。文献[17]通过分析不考虑节点同质性问题的 SFC 现有冗余备份部署方法，提出了一种异构备份部署方案。该方案可以确保备份服务器节点、虚拟网络功能（virtual network function，VNF）节点与原始节点之间的异构性，并根据该方案构建了具有异构备份的 SFC 部署模型。但由于未知漏洞的普遍性，攻击者仍能大范围利用漏洞攻破 SFC。

被动的防御方案无法有效防御基于未知漏洞的攻击方式，而主动防御则是一类不依赖攻击行为先验知识的防御机制，主要有移动目标防御（moving target defense，MTD）[18]和拟态安全防御两方面。其中，MTD 利用动态性、随机性、多样性思想对多个层面进行改造[19]，增加了系统环境的不确定性，从而加大攻击难度。文献[20]通过静态代码分析对指令操作数分类并与随机掩码进行异或操作，从数据层面提升了攻击者的漏洞利用难度，但该方法计算复杂，从设计层面实现开销较大。在拟态安全防御方面，为了建立基于“有毒带菌”的软/硬件设备的高安全性系统，邬江兴院士带领团队首创了“拟态安全防御”概念，设置多个功能等价的异构冗余执行体共同处理相同的数据，并对执行体进行基于负反馈的动态调度，以动态性、异构性、冗余性代替被动防御的静态性、相似性、单一性，从而达到系统安全风险可控的要求。

异构执行体的动态调度是拟态防御架构中最重要的环节，它负责构建当前的执行体服务集，实现执行体组件的动态变化，使整个架构保持异构冗余特性。因此，调度算法决定了架构的安全质量。目前，拟态调度方法的研究主要在于保证拟态系统的高异构性以及探测攻击并动态调整调度方法两个方面。文献[21]提出了一种完全随机的调度算法，通过伪随机数确定异构执行体的冗余度和编号并进行调度，但该方法的随机性使防御效果并不稳定。文献[22]提出了一种基于执行体异构度的调度方法，每次调度都尽可能选取与在线执行体集合相比异构度大的执行体，以确保拟态系统的异构性保持较高水平，降低了执行体间的共同漏洞，增大了攻击者的攻击代价与难度。文献[23]提出将 NFV 中 SFC 的攻防过程建模为斯坦科尔伯格博弈模型，通过动态地轮换存在安全威胁的 VNF 节点，有效地提升了 SFC 的安全性。但不同于控制层面的拟态，MSFC 防御架构须考虑每个执行体的性能。若采用文献[22-23]的方法，可能出现几次调度选取的执行体速度都不满足网络负载要求的情况，从而导致更严重的安全问题。文献[24]提出了一种基于生物种群模型的负反馈调度方法，将不同类型的执行体和攻击者建模为不同的生物种群，结合生物种群模型中的生存力以及系统的负载量计算每个执行体的选取系数，并以此为依据选取新的执行体集合。该方法通过动态选择被攻击次数较少的执行体，提升了系统的安全性。但由于忽略了系统的异构性，只能有

效防御普通的静态攻击者，对存在记忆能力且能够对相近目标实施快速攻击的适应型攻击者有一定局限性[25]。对于这类攻击者，系统的异构性越小，则被成功攻击的概率越高。因此，本文基于判决反馈提出了一种能够保证系统异构性，且在线执行体集合可以随网络变化自适应调整的调度方法。

2 拟态服务功能链防御架构

MSFC 防御架构将以主动防御的方式确保 SF 执行体正常执行对应的服务功能，MSFC 防御架构如图 1 所示。

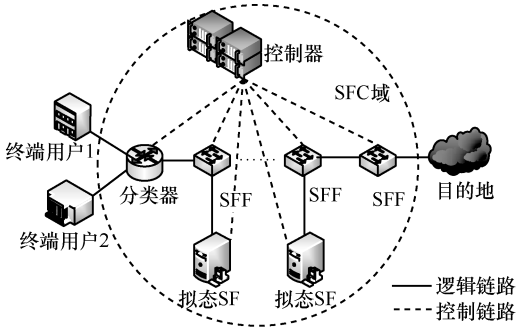

图 1 MSFC 防御架构

（1）SFF

具有转发功能的设备，负责根据网络服务头（network service header，NSH）信息对数据进行转发。

（2）控制器

负责定义 SFC，将流规则安装在 SFF 中以及将分类策略下发给分类器。

（3）分类器

负责根据控制层下发的分类策略给数据包加上相应的 NSH 信息，使数据包进入 SFC 域后可以按照规定的服务功能路径转发。

（4）拟态 SF

拟态 SF 组成如图 2 所示，包括调度器、分发器、缓存器、判决反馈器、控制分析器以及异构 SF 执行体池 6 个部分。

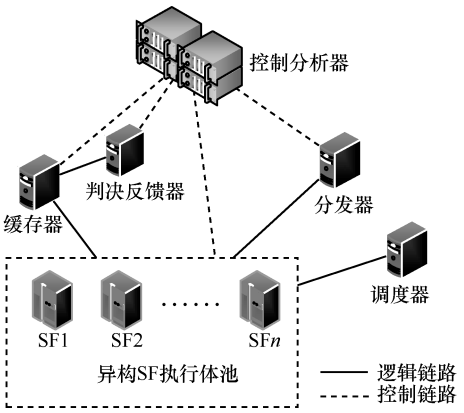

图 2 拟态 SF 组成

MSFC 防御架构工作流程如图 3 所示。

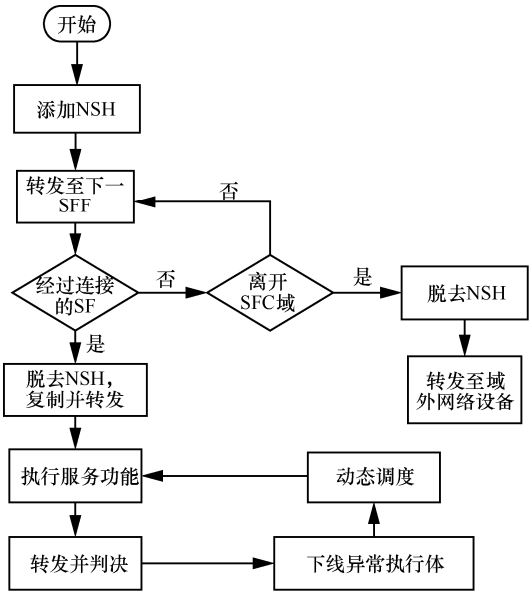

图 3 MSFC 防御架构工作流程

步骤 1 进入 SFC 域的数据包将先到达分类器，分类器依据控制层下发的分类策略为数据包加上对应的 NSH 信息，根据 NSH 信息将数据包转发至下一 SFF。

步骤 2 SFF 收到数据包后依据 NSH 信息判断是否需要经过它所连接的 SF，需要则进行步骤 4，反之则进行步骤 3。

步骤 3 判断数据包是否将离开 SFC 域，离开则 SFF 脱掉数据包的 NSH 并将其转发至 SFC 域外的网络设备；反之则直接转发至下一 SFF，回到步骤 2。

步骤 4 将数据包转发至分发器，分发器去

除其 NSH，将数据包复制多份并转发给各个在线 SF 执行体。执行体执行相应服务功能后将执行结果依次转发给判决反馈器。

步骤 5　控制分析器根据判决反馈器发送的判决结果下线异常 SF 执行体。

步骤 6　调度器根据执行体动态调度方法，每隔一定的调度时间从异构 SF 执行体池中选择异构度大的未上线 SF 执行体上线并将所有在线 SF 执行体下线。

MSFC 防御架构以 SFC 架构[1]为基础，参考了 DHR 模型，通过引入异构冗余 SF 执行体池，用多种异构 SF 执行体同时执行服务功能的方式代替传统由单一 SF 执行体执行服务功能的模型，保证系统的可靠性；通过基于判决反馈的执行体动态调度方法对 SF 执行体进行动态调度，保证系统的安全性；通过对异常 SF 执行体的及时下线，保证系统的内部纯洁性。其中，异构执行体的动态调度是拟态防御架构中最重要的环节，调度算法决定了架构的安全质量。因此，本文基于架构进一步提出了执行体动态调度方法以提升安全性。

3　执行体动态调度方法

3.1　符号和定义

本节给出执行体动态调度方法相关符号和定义。

定义 1　异构因子比较函数。设执行体池共有 m 个执行体 $\mathrm{SF}=\{\mathrm{SF}_1,\mathrm{SF}_2,\cdots,\mathrm{SF}_m\}$，每个 SF_i 由各种硬件、操作系统和软件等 e 种异构元素组成 $\mathrm{SF}_i=(\mathrm{sf}_i^1,\mathrm{sf}_i^2,\cdots,\mathrm{sf}_i^e)$，每个异构元素划分为可以比较的最小单位 $\mathrm{sf}_i^g=\{x_{i1}^g,x_{i2}^g,\cdots,x_{il_g}^g\}$，其中，$I_g$ 表示 SF_i 第 g 类异构元素的异构因子种类数。定义异构因子比较函数为：

$$c_0(x_{ik}^g,x_{jk}^g)=\begin{cases}0, & x_{ik}^g=x_{jk}^g\\ 1, & x_{ik}^g\neq x_{jk}^g\end{cases} \tag{1}$$

定义 2　单个执行体间异构度。根据定义 1，定义 SF_i 与 SF_j 的异构度 $\alpha(\mathrm{SF}_i,\mathrm{SF}_j)$ 为：

$$\alpha(\mathrm{SF}_i,\mathrm{SF}_j)=\sum_{g=1}^{e}\sum_{k=1}^{l_g}\delta_g\delta_{gk}c_0(x_{ik}^g,x_{jk}^g) \tag{2}$$

其中，用 δ_g 表示第 g 类异构元素的权值，δ_{gk} 表示第 g 类异构元素第 k 类异构因子的权值，从而区分不同种类异构元素以及单个异构元素中不同种类异构因子的关键性。

定义 3　整体异构度。根据定义 2，设有 n 个在线执行体 $\mathrm{SFo}=\{\mathrm{SFo}_1,\mathrm{SFo}_2,\cdots,\mathrm{SFo}_n\}$，定义 SFo 的整体异构度为：

$$\rho=\sum_{i=1}^{n}\sum_{j=i+1}^{n}\alpha(\mathrm{SFo}_i,\mathrm{SFo}_j) \tag{3}$$

定义 4　置信度。定义置信度集合 $B=\{b_{\mathrm{SF}_1},b_{\mathrm{SF}_2},\cdots,b_{\mathrm{SF}_m}\}$ 表示 SF 执行体的可信赖程度，每个 SF 执行体的置信度都会随着实际网络情况的变化而改变，集合中所有元素的初始值相同。

定义 5　执行速度影响值。定义集合 $U=\{u_{\mathrm{SF}_1},u_{\mathrm{SF}_2},\cdots,u_{\mathrm{SF}_m}\}$，$u_{\mathrm{SF}_i}$ 表示 SF_i 在执行体池中的相对执行速度。定义变量 $\varepsilon u_{\mathrm{SF}_i}$ 表示 SF_i 的执行速度影响值，依据网络流量变化更新，且 $\varepsilon>0$。

定义 6　系统负载及异常执行体总数增长量。定义 Δd_l 与 Δd_{a} 分别代表系统负载量和异常执行体总数较前一个调度周期的增长量。在调度周期内，若 Δd_l 与 Δd_{a} 同时极大增长，则 SF 执行体缓存溢出概率较大；若 Δd_l 无明显变化，Δd_{a} 极大增长，则 SF 执行体被攻击成功概率较大。上述情况均会造成输出的执行结果异常。

3.2　调度过程

设执行体池由 l 种执行不同类型服务功能的 SF 执行体组成 $\mathrm{SF}=\{S_1,S_2,\cdots,S_l\}$，假设下一个调度周期需要第 i 类服务功能，当 SFo 为前一个周期的在线执行体集合时，设置集合 W 表示在集合 S_i 中，所有不属于 SFo 的元素的集合。

（1）计算异构度

实际中受异构元素种类的限制，为了选出最佳 SF 执行体集合，加入异构元素间相对异构度的

影响。对于 $\forall \mathrm{SF}_i \in U$，计算 SF_i 与集合 SFo 中所有元素的异构度 $\gamma_i(\mathrm{SF}_i,\mathrm{SFo})$，计算式如下：

$$\gamma_i(\mathrm{SF}_i,\mathrm{SFo})=\frac{1}{n}\sum_{j=1}^{n}\alpha(\mathrm{SF}_i,\mathrm{SFo}_j) \tag{4}$$

（2）更新置信度

运行环境不同，SF 执行体池中的每个异构 SF 执行体的防御能力也有所差异。调度器需依据网络环境变化实时更新执行体置信度，并选择较难被成功攻击的 SF 执行体。

若调度器接收 SF_i 为异常 SF 执行体，则立即将其置信度设置为 0。比较集合 SF 中所有元素与 SF_i 的异构度，根据异构度对 SF 执行体的置信度进行更新。设置 δ_n 为置信度更新因子，且 $\delta_n>0$。由定义 2 可知，两个 SF 执行体间的异构度最大值 $\alpha_{\max}=\sum_{g=1}^{e}\sum_{k=1}^{l_g}\delta_g\delta_{gk}$。设置 $\alpha'=\frac{\alpha_{\max}}{e}$ 为异构度区分边界，则对集合 SF 中每个 SF_j 的置信度作如下更新：

$$(b_{\mathrm{SF}_j})_{\mathrm{new}}=\begin{cases}\dfrac{(b_{\mathrm{SF}_j})_{\mathrm{old}}}{\delta_n}\alpha(\mathrm{SF}_i,\mathrm{SF}_j)\leqslant\alpha'\\ \delta_n(b_{\mathrm{SF}_j})_{\mathrm{old}}\alpha(\mathrm{SF}_i,\mathrm{SF}_j)>\alpha'\end{cases} \tag{5}$$

若调度器接收 SF_i 为正常 SF 执行体，则对集合 SF 中每个 SF_j 的置信度作如下更新：

$$(b_{\mathrm{SF}_j})_{\mathrm{new}}=\begin{cases}(b_{\mathrm{SF}_j})_{\mathrm{old}}+\delta_n\alpha(\mathrm{SF}_i,\mathrm{SF}_j)\leqslant\alpha'\\ (b_{\mathrm{SF}_j})_{\mathrm{old}}\alpha(\mathrm{SF}_i,\mathrm{SF}_j)>\alpha'\end{cases} \tag{6}$$

（3）计算执行速度影响值

实际网络流量大小在不同时段是不同的。在流量使用高峰时段，执行速度慢的 SF 执行体即使未被成功攻击，也可能出现缓存队列的溢出使其输出结果与正常 SF 执行体不匹配，因而被判定为异常 SF 执行体。当大部分在线 SF 执行体的执行速度不满足网络要求时，系统甚至可能直接崩溃。因此，调度器需要考虑 SF 执行体的执行速度，即调度算法必须能随网络流量的变化进行适当调整。

在 MSFC 防御架构中，数据包先到达分发器，再由分发器复制并转发给各个 SF 执行体，因此可以将分发器的负载量变化作为实际网络流量的变化。设在一个调度周期内，分发器的总负载量增加 c，则执行速度影响值计算如下：

$$\varepsilon_{\mathrm{new}}=\varepsilon_{\mathrm{old}}e^{c} \tag{7}$$

（4）计算可信赖度并更新执行速度影响值

每次调度都根据需要的服务功能类型确定从哪个集合选取 SF 执行体。假设选取的集合为 S_k，对于 $\forall \mathrm{SF}_i \in S_k$，根据调度过程（1）计算的异构度、调度过程（2）计算的置信度和调度过程（3）计算的执行速度影响值，可计算 SF_i 的基础选择度如下：

$$\sigma_i=\frac{b_{\mathrm{SF}_i}(\gamma_i(\mathrm{SF}_i,\mathrm{SFo})+\varepsilon u_{\mathrm{SF}_i})}{\rho} \tag{8}$$

根据定义 6，若 SF 执行体缓存溢出，则需增大执行速度对可信赖度计算的影响；若 SF 执行体被攻击者成功攻击，则需增大置信度和异构度对可信赖度计算的影响。设置系统负载量与异常执行体个数的增长阈值分别为 ξ_1 与 ξ_a，且将 Δd_l 与 Δd_a 超出增长阈值的情况都视作极大增长。对于 $\forall \mathrm{SF}_i \in S_k$，$\mathrm{SF}_i$ 的可信赖度计算方式如下：

$$\zeta_i=\begin{cases}\sigma_i\dfrac{\Delta d_l}{\xi_l}\varepsilon u_{\mathrm{SF}_i}\Delta d_a>\xi_a,\Delta d_l>\xi_l\\ \sigma_i\dfrac{\Delta d_a}{\xi_a}(b_{\mathrm{SF}_i}+\gamma_i(\mathrm{SF}_i,\mathrm{SFo}))\Delta d_a>\xi_a,\Delta d_l<\xi_l\\ \sigma_i,\text{其他}\end{cases} \tag{9}$$

进一步，对执行速度影响值 ε 进行如下更新：

$$\varepsilon_{\mathrm{new}}=\begin{cases}\dfrac{\Delta d_l}{\xi_l}\varepsilon_{\mathrm{old}}\Delta d_a>\xi_a,\Delta d_l>\xi_l\\ \dfrac{\xi_a}{\Delta d_a}\varepsilon_{\mathrm{old}}\Delta d_a>\xi_a,\Delta d_l<\xi_l\\ \varepsilon_{\mathrm{old}},\text{其他}\end{cases} \tag{10}$$

依据前后两个调度周期的系统负载量和异常执行体总数的变化，以基础选择度为基准计算下一调度周期 SF 执行体的可信赖度，能够使执行体调度与网络变化的结合更加紧密。

（5）选取执行体集合

假设在线 SF 执行体集合共需要 l 个 SF 执行体。调度器计算集合 W 中所有元素的可信赖度后，对其进行由高到低排序，选出可信赖度靠前的 $2l$ 个 SF 执行体形成新的 SF 执行体集合 $S_i^* = \{\mathrm{SF}_{d_1}, \mathrm{SF}_{d_2}, \cdots, \mathrm{SF}_{d_{2l}}\}$。为了提高所选在线 SF 执行体集合的整体异构度，以最大化攻击者的攻击难度，对 S_i^* 中所有 SF 执行体进行遍历，组合得到包含 l 个 SF 执行体的集合并计算其整体异构度。最后，选择整体异构度最大者作为新的在线 SF 执行体集合。

（6）更新调度周期

系统的安全性将随调度周期的减小而增加，但同时将带来更高的系统资源消耗。因为网络环境的变化，难以确定固定的调度时间 T_s 使系统资源消耗与安全性达到最佳平衡。

设置集合 $T = \{T_{s1}, T_{s2}, \cdots, T_{sk}\}, k \to +\infty$，其中，$T_{si}$ 表示第$(i-1)$次调度到第 i 次调度对应的时间间隔。考虑异常 SF 执行体信息最能体现网络变化，将异常 SF 执行体个数 r_i 作为 T_{si} 的调整依据。若 r_i 大于设定阈值 ξ_n，则认为当前系统需要通过降低调度时间以提升安全性；反之，则认为仅依靠调度算法的内部调节就可以降低异常 SF 执行体个数。设 T_{si} 的最大取值为 $\xi_{\max}$，τ 表示异常执行体个数为零的次数，从首次出现该情况开始累加，一旦出现个数不为零的情况则立即将 τ 置零，待异常执行体个数再次为零时重新开始累加，则对下一调度周期进行如下更新：

$$T_{si} = \begin{cases} \left(\dfrac{1}{r_{i-1} - \xi_n}\right) T_{s(i-1)} r_{i-1} > \xi_n \\ \left(1 + \ln(\tau + 1)\right) T_{s(i-1)} r_{i-1} = 0, T_{s(i-1)} < \dfrac{\xi_{\max}}{\left(1 + \ln(\tau + 1)\right)} \\ T_{s(i-1)}, \text{其他} \end{cases} \quad (11)$$

根据前一个调度周期内异常 SF 执行体的总数对调度时间动态调整，使得防御系统可以在合适的系统代价始终保持较高水准的安全性。

3.3　调度算法

调度算法根据 SF 执行体的可信赖度从异构 SF 执行体池中选择满足要求的 SF 执行体集合。因为影响 SF_i 可信赖度的所有因素都会随着网络变化而实时变化，所以 SF_i 的可信赖度可以动态更新，又因为 SF_i 与集合 SFo 的异构度是可信赖度计算影响因素之一，所以该调度方法既能保持动态性和随机性，又能保证选出的 SF 执行体集合与集合 SFo 维持一定水平的异构度。

定时地替换在线执行体，可以在保持系统异构性的同时降低每个执行体的暴露时间，系统结构信息的不确定性提升，漏洞被发现的风险降低。因为系统处于不断更新的状态且一直保持较高的异构度，所以可以清除适应型攻击者的前期努力及潜在漏洞。在发现被感染的异常执行体后，系统还可以通过调度直接将其下线，达到有效阻断攻击者对异常执行体持续控制的目的。

假设下一个调度周期需要第 i 类服务功能。具体见算法 1（符号沿用 4.2 节的标记）。

输入　在线 SF 执行体集合 SFo，执行第 i 类服务功能的 SF 执行体集合 S_i，执行速度影响因子 ε，SF 执行体间的相对执行速度集合 U，SF 中所有元素的置信度集合 B，系统负载量 Δd_l，异常执行体总数较前一个调度周期对应值的增长量 Δd_{a}

输出　下一个调度周期的执行体集合 $\mathrm{SFo}_{\mathrm{next}}$

$\mathrm{SFo}_{\mathrm{next}} = \varnothing$

temp=∅

计算 S_i 中非在线 SF 的基础可信赖度

if $\Delta d_{\mathrm{a}} > \xi_a \,\&\&\, \Delta d_l > \xi_l$

　for d in $(S_i - \mathrm{SFo})$

$$\zeta_d = \sigma_d \frac{\Delta d_l}{\xi_l} \varepsilon u_{\mathrm{SF}_d}$$

　put ζ_d to temp

$$\varepsilon_{\mathrm{new}} = \frac{\Delta d_l}{\xi_l} \varepsilon_{\mathrm{old}}$$

　end for

else if $\Delta d_a > \xi_a \&\& \Delta d_l < \xi_l$

for d in $(S_i - \mathrm{SFo})$

$$\zeta_d = \sigma_d \frac{\Delta d_a}{\xi_a}\left(b_{\mathrm{SF}_d} + \gamma_i(\mathrm{SF}_i, \mathrm{SFo})\right)$$

put ζ_d to temp

$$\varepsilon_{\mathrm{new}} = \frac{\xi_a}{\Delta d_a}\varepsilon_{\mathrm{old}}$$

end for

else

for d in $(S_i - \mathrm{SFo})$

$$\zeta_d = \sigma_d$$

put ζ_d to temp

end for

end if

$E_{\mathrm{a}} = \mathrm{sort}(\mathrm{temp})$

可信赖度前 $2l$ 的 SF 组成集合 E_s

$E_{\mathrm{b}} = \mathrm{getalllist}(E_s, l)$

$\mathrm{SFo}_{\mathrm{next}} = E_{\mathrm{b}}(0)$

for i in $\mathrm{len}(E_{\mathrm{b}})$

if $\left(\rho(\mathrm{SFo}_{\mathrm{next}}) > \rho\left(E_{\mathrm{b}}(i)\right)\right)$

$\mathrm{SFo}_{\mathrm{next}} = E_{\mathrm{b}}(i)$

end if

end for

return $\mathrm{SFo}_{\mathrm{next}}$

4 仿真分析

4.1 仿真环境

本节对在数据转发层设计的调度方法进行仿真分析，测试其有效性。通过 Open vSwitch 虚拟交换机实现分类器与 SFF，数据层拓扑如图 4 所示，同时将物理计算机以物理连接的方式分别连接到分类器、SFF3 所挂载的物理接口以模拟源目的特征组，最后在物理服务器安装多台虚拟机以构建拟态 SF。虚拟机集合表示如下：

$$\mathrm{V} = \{\mathrm{V}_1, \mathrm{V}_2, \cdots, \mathrm{V}_{12}\} \tag{12}$$

V_1 及 V_2 安装 CentOS 操作系统且设置 V_1 完成控制分析器功能，V_2 完成分发器、缓存器、判

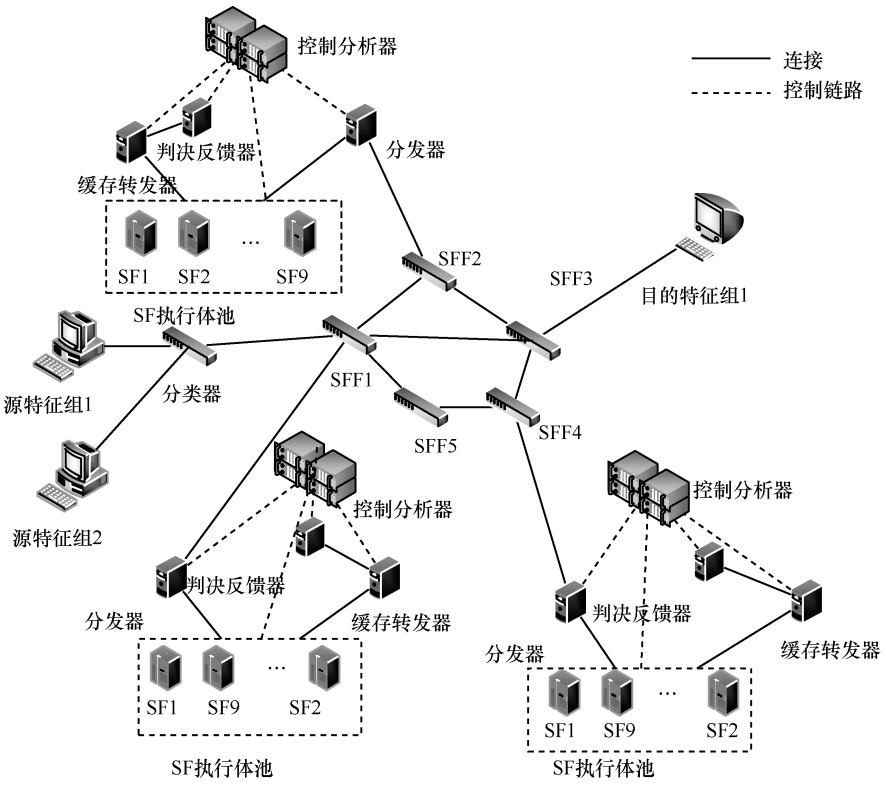

图 4 数据层拓扑

决反馈器功能。V_3～V_{12} 设置为防火墙与负载均衡的 SF 执行体池，其中，V_3～V_7 安装不同操作系统与防火墙组合，V_8～V_{12} 安装不同操作系统与负载均衡组合，SF 执行体池构建见表 2。将服务器中所有虚拟机设置相同的虚拟机端口组，将拟态 SF 加入 SFC 中，从而完成实验平台的搭建。

表 2　SF 执行体池构建

虚拟机	操作系统	软件
V_3	Ubuntu	iptables
V_4	Windows	Kaspersky Anti-Hacker
V_5	CentOS	XELIOS Personal Firewall
V_6	Windows	ZoneAlarm Pro
V_7	Windows	Norton Personal Firewall
V_8	CentOS	LVS
V_9	Windows	Nginx
V_{10}	CentOS	HAproxy
V_{11}	Ubuntu	LVS
V_{12}	Ubuntu	Nginx

使用 Java 语言定义不同异构元素的 SF 执行体类，同时针对不同的异构元素设置对应的漏洞类，将对应的漏洞对象作为 SF 执行体类的成员变量。当 SF 执行体类实例化时，赋予不同类型的 SF 执行体对象不同的最大可承受负载量，即在不发生异常的情况下设备能够处理的最大数据量，以完成异构 SF 执行体集合的构建。

4.2　调度方法安全性测试

为验证本文调度算法能在网络负载量增长时选取合适的执行体集合，假设系统初始调度时间为 T，在时间间隔 $5T$ 内，分别用随机调度算法[21]、异构度调度算法[22]以及本文的判决反馈动态调度算法从异构执行体池中获取执行体集合。将获取的执行体集合中所有不满足负载量要求的执行体视为异常执行体，得到异常执行体的总数与负载量关系。依据获取的数据得到实验仿真结果，异常执行体的总数与负载量关系如图 5 所示。

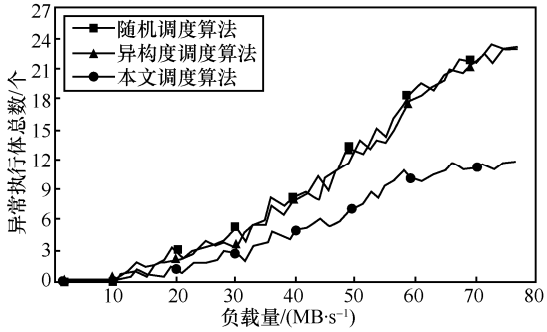

图 5　异常执行体的总数与负载量关系

由仿真结果可知，随着负载量的不断增大，使用随机调度算法与异构度调度算法的系统异常执行体总数明显大于使用判决反馈动态调度算法的系统异常执行总数，且差距随着负载量的增大而增大。这是由于当系统负载量较小时，大多数 SF 执行体都满足系统负载要求，此时 3 种算法系统的异常执行体总数差距不大。随着系统负载量的增加，不满足系统负载要求的 SF 执行体不断增多，使用判决反馈动态调度算法的系统相较于另两种能尽量避免选择不符合负载要求的 SF 执行体，减少异常执行体总数。

为验证判决反馈动态调度算法能在网络负载量急剧变化时及时做出相应调整，设置初始负载量为所有 SF 执行体都能满足的值，并以 $5T$ 为测量间隔进行仿真。设定负载量在第二个调度周期内发生唯一的一次增长，记录使用 3 种不同的调度算法时，异常执行体的总数与负载量增长值的关系，并依据获取的数据得到实验仿真结果，异常执行体的总数与负载量增长值的关系如图 6 所示。

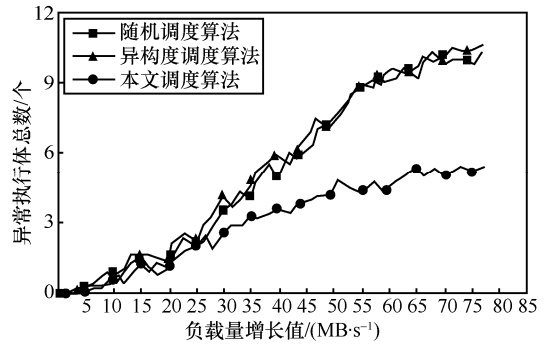

图 6　异常执行体的总数与负载量增长值的关系

仿真结果表明，当负载量增长值在 25 MB/s 以内时，使用 3 种调度算法的系统异常执行体总数相差无几；当负载量增长值大于 25 MB/s 时，使用随机调度算法与异构度调度算法的系统异常执行体总数的增速呈上升趋势，而使用判决反馈动态调度算法的系统异常执行体总数的增速呈下降趋势。因为当负载量增长值在 25 MB/s 以内时，未达到 $\Delta d_a > \xi_a$、$\Delta d_l > \xi_l$ 的条件，此时本文提出的调度算法并未受到执行速度影响因子的干涉，且此时执行体池中不符合负载要求的执行体不多，所以 3 种调度算法对应系统的异常执行体总数相差不多。当负载增长值大于 25 MB/s 时，满足 $\Delta d_a > \xi_a$、$\Delta d_l > \xi_l$ 的条件，此时判决反馈动态调度算法将受到执行速度影响因子的干涉，进而选取执行速度较快的 SF 执行体，因此异常执行体总数的增速相对下降。而其他两种调度算法未将负载值增量作为影响因素，且执行体池中不符合负载要求的执行体逐渐增多，因此异常执行体总数的增速必然相对增加。

为验证判决反馈动态调度算法能够适应攻击者不同强度的攻击，通过模拟适应型攻击者的攻击模式，以 5T 为测量时间间隔，以不同的攻击频率模拟实际网络中攻击者不同的攻击强度，对运行上述 3 种调度算法的系统进行模拟攻击。最终得到系统被成功攻击的概率与攻击强度的关系，并依据获取的数据得到实验仿真结果，系统被成功攻击的概率与攻击强度关系如图 7 所示。

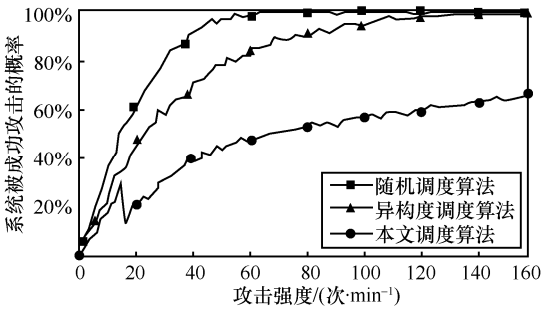

图 7　系统被成功攻击的概率与攻击强度关系

仿真结果表明，随机调度算法与异构度调度算法的系统被成功攻击的概率随着攻击强度的增强而不断增加。异构度调度算法将与前一调度周期的在线 SF 执行体集合间的异构度作为调度依据，所以该算法对应的系统被成功攻击的概率上升较慢。当攻击强度小于 15 次/min，未达到动态调整调度时间的触发条件时，判决反馈动态调度算法考虑了与前一调度周期的在线 SF 执行体集合间的异构度以及与异常执行体的异构度，所以该算法的系统被成功攻击的概率略小于异构度算法；攻击强度大于 15 次/min 时，该算法动态减小调度时间，以花费更多系统资源的代价提升系统安全性，所以系统被成功攻击的概率瞬间锐减且该概率的上升速率明显降低。

4.3　调度方法效率测试

系统被成功攻击的概率与系统资源的总调度花费成反比，针对同一段系统运行时间 t，若将系统调度时间 T_s 设置为较大值，则系统被成功攻击的平均概率较高但是总调度花费较低；反之系统被成功攻击的平均概率较低但是总调度花费较高。为验证判决反馈动态调度算法能依据网络环境的变化适当的调整系统调度时间，通过模拟适应型攻击者的攻击模式，以不同的攻击频率模拟实际网络中攻击者不同的攻击强度，对运行在同种调度方法下设置调度时间分别固定为 T=3 s，T=9 s 以及设置动态调度时间在 3～9 s 的系统进行了模拟攻击。最终得到系统被成功攻击的概率与攻击强度的关系，并依据获取的数据得到实验仿真结果，系统被成功攻击的概率与攻击强度的关系如图 8 所示。同样地，得到不同调度时间系统平均调度资源花费关系如图 9 所示。

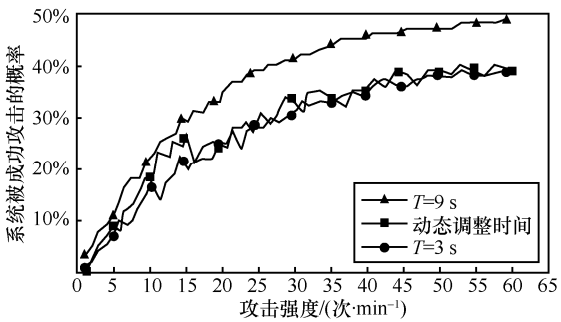

图 8　系统被成功攻击的概率与攻击强度的关系

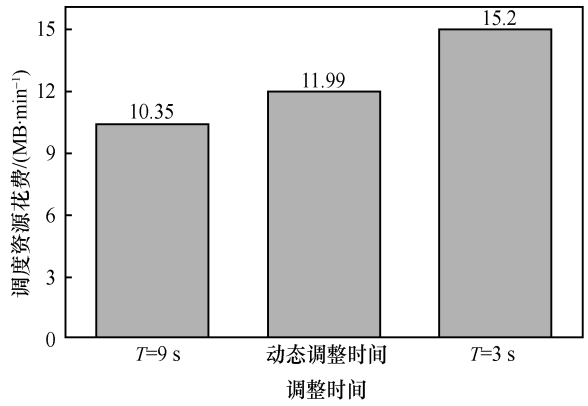

图 9　不同调度时间系统平均调度资源花费

图 8 显示，随着攻击强度的增长，系统被成功攻击的概率不断上升。当调度时间为 9 s 时，系统被成功攻击的概率明显大于动态选取调度时间和调度时间为 3 s 的概率，且差距随着攻击强度增加不断增大。调度时间在 3～9 s 动态调整，则该系统被成功攻击的概率介于调度时间为 3～9 s 的系统，且与前者系统十分接近。图 9 显示，动态选取调度时间与调度时间为 9 s 的系统平均调度资源花费较低，调度时间为 3 s 的系统平均调度资源花费较高。实验结果表明，本文根据异常 SF 执行体个数动态调整系统调度时间的方法可以使系统在保持一定安全性的同时尽量花费较小的调度代价。

5　结束语

面对 SFC 的安全性问题，本文提出了数据转发层面的 MSFC 防御架构，研究基于判决反馈的动态调度方法以判决器反馈的异常执行体信息、执行体的异构度以及系统的实际负载量作为调度影响因素，使得调度方法可以根据网络实际变化进行自适应调整。此外，该调度方法基于判决反馈对调度时间进行调整，以达到系统花费与安全性的最佳平衡，降低了系统的资源开销。仿真结果表明，该调度方法可以在平衡系统花费与安全性的基础上，选出更符合当前网络需求的高异构度执行体集合，从而提升了系统的安全性和可靠性并减少了系统开销。但本文提出的调度方法并未考虑执行体选取的时延问题。在后续研究中，将对该方法进行相关优化，尽可能在保持系统安全性和高效性的同时降低执行体选取时延，使之具有更好的实用性。

参考文献：

[1] RFC. Service function chaining (SFC) architecture[R]. 2015.

[2] 李天龙. 多域网络安全服务编排系统的设计与实现[D]. 北京: 北京交通大学, 2018.
LI T L. The design and implement of multi-domain network security service orchestration system[D]. Beijing: Beijing Jiaotong University, 2018.

[3] RFC. Forwarding and control element separation (ForCES) protocol specification[R]. 2010.

[4] BIFULCO R, CANONICO R, BRUNNER M, et al. A practical experience in designing an OpenFlow controller[C]// Proceedings of 2012 European Workshop on Software Defined Networking. Piscataway: IEEE Press, 2012: 61-66.

[5] Table of contents[EB]. 2006.

[6] JAIN R, PAUL S. Network virtualization and software defined networking for cloud computing: a survey[J]. IEEE Communications Magazine, 2013, 51(11): 24-31.

[7] GUERZONI R. Network functions virtualization an introduction, benefits, enablers, challenges and call for action[C]// SDN and OpenFlow World Congress, St Louis. [S.l.:s.n.], 2012.

[8] CHOWDHURY N M M K, BOUTABA R. A survey of network virtualization[J]. Computer Networks, 2010, 54(5): 862-876.

[9] 宋永春. 基于 SDN 的设备间服务链设计与实现[D]. 兰州: 兰州大学, 2017.
SONG Y C. The design and realization of equipment service-chain in SDN[D]. Lanzhou: Lanzhou University, 2017.

[10] IETF. Problem statement for service function chaining: RFC 7498[S]. 2018.

[11] OKTIAN Y E, LEE S, LEE H. Mitigating Denial of Service (DoS) attacks in OpenFlow networks[C]//Proceedings of 2014 International Conference on Information and Communication Technology Convergence (ICTC). Piscataway: IEEE Press, 2014: 325-330.

[12] JABEUR N, MOH A N S, BARKIA M M. A bully approach for competitive redundancy in heterogeneous wireless sensor network[J]. Procedia Computer Science, 2016(83): 628-635.

[13] 邬江兴. 网络空间拟态安全防御[J]. 保密科学技术, 2014(10): 1, 4-9.
WU J X. Cyberspace pseudo security defense [J]. Secrecy Science and Technology, 2014(10): 1, 4-9.

[14] 马海龙, 伊鹏, 江逸茗, 等. 基于动态异构冗余机制的路由

器拟态防御体系结构[J]. 信息安全学报, 2017, 2(1): 29-42.
MA H L, YI P, JIANG Y M, et al. Dynamic heterogeneous redundancy based router architecture with mimic defenses[J]. Journal of Cyber Security, 2017, 2(1): 29-42.

[15] 马海龙, 江逸茗, 白冰, 等. 路由器拟态防御能力测试与分析[J]. 信息安全学报, 2017, 2(1): 43-53.
MA H L, JIANG Y M, BAI B, et al. Tests and analyses for mimic defense ability of routers[J]. Journal of Cyber Security, 2017, 2(1): 43-53.

[16] DURANTE L, SENO L, VALENZA F, et al. A model for the analysis of security policies in service function chains[C]//Proceedings of 2017 IEEE Conference on Network Softwarization (NetSoft). Piscataway: IEEE Press, 2017.

[17] XIE J C, YI P, ZHANG Z, et al. A service function chain deployment scheme based on heterogeneous backup[C]//Proceedings of 2018 IEEE 18th International Conference on Communication Technology (ICCT). Piscataway: IEEE Press, 2018.

[18] JAJODIA S, GHOSH A K, SWARUP V, et al. Moving Target Defense[M]. New York, NY: Springer New York, 2011.

[19] OKHRAVI H, HOBSON T, BIGELOW D, et al. Finding focus in the blur of moving-target techniques[J]. IEEE Security & Privacy, 2014, 12(2): 16-26.

[20] CADAR C, AKRITIDIS P, COSTA M, et al. Data randomization[R]. 2008.

[21] 邬江兴, 李军飞. 一种异构功能等价体调度装置及方法: CN106161417A [P]. 2016.
WU J X, LI J F. A heterogeneous functional equivalent scheduling device and method: CN106161417A [P]. 2016.

[22] QI C, WU J X, HU H C, et al. Dynamic-scheduling mechanism of controllers based on security policy in software-defined network[J]. Electronics Letters, 2016, 52(23): 1918-1920.

[23] 季新生, 徐水灵, 刘文彦, 等. 一种面向安全的虚拟网络功能动态异构调度方法[J]. 电子与信息学报, 2019, 41(10): 2435-2441.
JI X S, XU S L, LIU W Y, et al. A security-oriented dynamic and heterogeneous scheduling method for virtual network function[J]. Journal of Electronics & Information Technology, 2019, 41(10): 2435-2441.

[24] 王禛鹏. 拟态网络操作系统调度与裁决机制研究及实现[D]. 郑州: 战略支援部队信息工程大学, 2017.
WANG Z P. Research on the scheduling and decision-making mechanism of mimic network operating system[D]. Zhengzhou: Information Engineering University, 2017.

[25] 王宇. 网络主动防御与主动防御网络[J]. 保密科学技术, 2014(11): 27-34.
WANG Y. Cyber active defense and active defense network [J]. Secrecy Science and Technology, 2014(11): 27-34

[作者简介]

李传煌（1980− ），男，博士，浙江工商大学教授、硕士生导师，主要研究方向为软件定义网络、开放可编程网络、边缘计算、人工智能应用等。

唐晶晶（1998− ），女，浙江工商大学硕士生，主要研究方向为软件定义网络、人工智能应用。

陈泱婷（1998− ），女，浙江工商大学硕士生，主要研究方向为软件定义网络、人工智能应用。

雷睿（1996− ），男，浙江工商大学硕士生，主要研究方向为软件定义网络、人工智能应用。

陈超（1986− ），男，博士，浙江工商大学副教授、硕士生导师，主要研究方向为下一代无线通信网络技术、网络编码、机器/深度学习等。

王伟明（1964− ），男，博士，浙江工商大学教授、硕士生导师，主要研究方向为新一代网络架构、开放可编程网络。